Signal Analysis: Wavelets, Filter Banks, Time-Frequency Transforms and Applications. Alfred Mertins

Print ISBN 0-471-98626-7 Electronic ISBN 0-470-84183-4

Chapter 4

Examples of Discrete Transforms

In this chapter we discuss the most important fixed discrete transforms. We start with the z-transform, which is a fundamental tool for describing the input/output relationships in linear time-invariant (LTI) systems. Then we discuss several variants of Fourier series expansions, namely the discrete-time Fourier transform, the discrete Fourier transform (DFT), and the fast Fourier transform (FFT). The remainder of this chapter is dedicated to other discrete transforms that are of importance in digital signal processing, such as the discrete cosine transform, the discrete sine transform, the discrete Hartley transform, and the discrete Hadamard and Walsh–Hadamard transform.

4.1 The z-Transform

The z-transform of a discrete-time signal $x(n)$ is defined as

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)\, z^{-n}. \tag{4.1}$$

Note that the time index n is discrete, whereas z is a continuous parameter. Moreover, z is complex, even if $x(n)$ is real. Further note that for $z = e^{j\omega}$ the z-transform (4.1) is equal to the discrete-time Fourier transform.

In general, convergence of the sum in (4.1) depends on the sequence $x(n)$ and the value z. For most sequences we only have convergence in a certain region of the z-plane, called the *region of convergence* (ROC). The ROC can be determined by finding the values r for which

$$\sum_{n=-\infty}^{\infty} \left| x(n)\, r^{-n} \right| < \infty. \tag{4.2}$$

Proof. With $z = r\, e^{j\phi}$ we have

$$\begin{aligned} |X(z)| &= \left| \sum_{n=-\infty}^{\infty} x(n)\, z^{-n} \right| \\ &= \left| \sum_{n=-\infty}^{\infty} x(n)\, r^{-n}\, e^{-j\phi n} \right| \\ &\leq \sum_{n=-\infty}^{\infty} \left| x(n)\, r^{-n}\, e^{-j\phi n} \right| \\ &= \sum_{n=-\infty}^{\infty} \left| x(n)\, r^{-n} \right|. \end{aligned} \tag{4.3}$$

Thus, $|X(z)|$ is finite if $x(n)r^{-n}$ is absolutely summable. □

The inverse z-transform is given by

$$x(n) = \frac{1}{j2\pi} \oint_C X(z)\, z^{n-1} dz. \tag{4.4}$$

The integration has to be carried out counter-clockwise on a closed contour C in the complex plane, which encloses the origin and lies in the region of convergence of $X(z)$.

Proof of (4.4). We multiply both sides of (4.1) with z^{k-1} and integrate over a closed contour in a counter-clockwise manner:

$$\begin{aligned} \oint_C X(z) z^{k-1} dz &= \oint_C \sum_{n=-\infty}^{\infty} x(n) z^{k-1-n} dz \\ &= \sum_{n=-\infty}^{\infty} x(n) \oint_C z^{k-1-n} dz. \end{aligned} \tag{4.5}$$

Invoking the *Cauchy integral theorem*

$$\frac{1}{2\pi j} \oint_C z^{k-1-n} dz = \delta_{nk}. \tag{4.6}$$

finally yields (4.4). □

Reconstruction formulae simpler than (4.4) can be found for rational $X(z)$, that is for

$$X(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} \ \ldots}{a_0 + a_1 z^{-1} + a_2 z^{-2} \ \ldots}.$$

Methods based on the residue theorem, on partial fraction expansion, and on a direct expansion of $X(z)$ into a power series in z^{-1} are known. For more detail, see e.g. [80, 113].

The simplest example is the z-transform of the discrete impulse:

$$\delta(n) = \begin{cases} 1, & n = 0 \\ 0, & \text{otherwise.} \end{cases}$$

We have

$$\delta(n) \longleftrightarrow \sum_{n=-\infty}^{\infty} \delta(n)\, z^{-n} = 1. \tag{4.7}$$

For a delayed discrete impulse it follows that

$$\delta(n - n_0) \longleftrightarrow \sum_{n=-\infty}^{\infty} \delta(n - n_0)\, z^{-n} = z^{-n_0}. \tag{4.8}$$

In the following, the most important properties of the z-transform will be briefly recalled. Proofs which follow directly from the definition equation of the z-transform are omitted.

Linearity

$$v(n) = \alpha x(n) + \beta y(n) \longleftrightarrow V(z) = \alpha X(z) + \beta Y(z). \tag{4.9}$$

Convolution

$$v(n) = x(n) * y(n) \longleftrightarrow V(z) = X(z)\, Y(z). \tag{4.10}$$

Proof.

$$\begin{aligned} V(z) &= \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x(k)\, y(n-k)\, z^{-n} \\ &= \sum_{k=-\infty}^{\infty} x(k)\, z^{-k} \sum_{m=-\infty}^{\infty} y(m)\, z^{-m} \\ &= X(z)\, Y(z). \end{aligned}$$

Shifting

$$x(n-n_0) \longleftrightarrow z^{-n_0}\, X(z). \tag{4.11}$$

This result is obtained by expressing $v(n)$ as $v(n) = x(n) * \delta(n-n_0)$ and using the convolution property above.

Scaling/Modulation. For any real or complex $a \neq 0$, we have

$$a^n\, x(n) \longleftrightarrow X\left(\frac{z}{a}\right). \tag{4.12}$$

This includes $a = e^{j\omega}$ such that

$$e^{j\omega n}\, x(n) \longleftrightarrow X\left(e^{-j\omega} z\right). \tag{4.13}$$

Time Inversion

$$x(-n) \longleftrightarrow X\left(\frac{1}{z}\right). \tag{4.14}$$

Derivatives

$$n\, x(n) \longleftrightarrow -z\, \frac{d\, X(z)}{dz}. \tag{4.15}$$

Proof.

$$\begin{aligned} -z\, \frac{d\, X(z)}{dz} &= -z \frac{d}{dz} \sum_{n=-\infty}^{\infty} x(n)\, z^{-n} \\ &= \sum_{n=-\infty}^{\infty} n\, x(n)\, z^{-n} \\ &\updownarrow \\ &n\, x(n). \end{aligned}$$

Conjugation. Given the correspondence $x(n) \longleftrightarrow X(z)$, we have

$$x^*(n) \longleftrightarrow X^*(z^*). \tag{4.16}$$

Proof.

$$\begin{aligned} X^*(z^*) &= \left(\sum_{n=-\infty}^{\infty} x(n) \, [z^*]^{-n} \right)^* \\ &= \left(\sum_{n=-\infty}^{\infty} x(n) \, [z^{-n}]^* \right)^* \\ &= \sum_{n=-\infty}^{\infty} x^*(n) \, z^{-n}. \end{aligned}$$

Paraconjugation. Given the correspondence $x(n) \longleftrightarrow X(z)$, we have

$$x^*(-n) \longleftrightarrow \tilde{X}(z), \qquad \text{where} \qquad \tilde{X}(z) = [X(z)]^* \, |_{|z|=1}. \tag{4.17}$$

That is, $\tilde{X}(z)$ is derived from $X(z)$ by complex conjugation on the unit circle.

Proof.

$$\begin{aligned} \tilde{X}(z) &= \left[\sum_k x(k) z^{-k} \right]^* |_{|z|=1} \\ &= \sum_k x^*(k) z^k \\ &= \sum_n x^*(-n) z^{-n} \\ &\quad \updownarrow \\ &\quad x^*(-n). \end{aligned} \tag{4.18}$$

For real signals $x(n)$, it follows that

$$x(-n) \longleftrightarrow \tilde{X}(z) = X(z^{-1}). \tag{4.19}$$

Multiplication with $\cos \omega n$ and $\sin \omega n$. If $x(n) \longleftrightarrow X(z)$, then

$$\cos \omega n \; x(n) \longleftrightarrow \frac{1}{2} \left[X(e^{j\omega} z) + X(e^{-j\omega} z) \right] \tag{4.20}$$

and

$$\sin \omega n \; x(n) \longleftrightarrow \frac{j}{2} \left[X(e^{j\omega} z) - X(e^{-j\omega} z) \right]. \tag{4.21}$$

This follows directly from (4.13) by expressing $\cos \omega n$ and $\sin \omega n$ via Euler's formulae $\cos \alpha = \frac{1}{2}[e^{j\alpha} + e^{-j\alpha}]$ and $\sin \alpha = \frac{j}{2}[e^{j\alpha} - e^{-j\alpha}]$.

Multiplication in the Time Domain. Let $x(n)$ and $y(n)$ be real-valued sequences. Then

$$v(n) = x(n)\, y(n) \longleftrightarrow V(z) = \frac{1}{2\pi j} \oint_C X(\nu) Y\Big(\frac{z}{\nu}\Big)\, \nu^{-1}\, d\nu, \tag{4.22}$$

where C is a closed contour that lies within the region of convergence of both $X(z)$ and $Y(\frac{1}{z})$.

Proof. We insert (4.4) into

$$V(z) = \sum_{n=-\infty}^{\infty} x(n)\, y(n)\, z^{-n}. \tag{4.23}$$

This yields

$$\begin{aligned} V(z) &= \sum_{n=-\infty}^{\infty} \left[\frac{1}{j2\pi} \oint_C X(\nu)\, \nu^{n-1} d\nu \right] y(n)\, z^{-n} \\ &= \frac{1}{j2\pi} \oint_C X(\nu) \underbrace{\left[\sum_{n=-\infty}^{\infty} y(n) \Big(\frac{z}{\nu}\Big)^{-n} \right]}_{Y(\frac{z}{\nu})} \nu^{-1}\, d\nu \end{aligned} \tag{4.24}$$

□

Using the same arguments, we may write for complex-valued sequences

$$v(n) = x(n)\, y^*(n) \longleftrightarrow V(z) = \frac{1}{2\pi j} \oint_C X(\nu) Y^*\Big(\frac{z^*}{\nu^*}\Big)\, \nu^{-1}\, d\nu. \tag{4.25}$$

4.2 The Discrete-Time Fourier Transform

The discrete-time Fourier transform of a sequence $x(n)$ is defined as

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)\, e^{-j\omega n}. \tag{4.26}$$

Due to the 2π-periodicity of the complex exponential, $X(e^{j\omega})$ is periodic: $X(e^{j\omega}) = X(e^{j(\omega + 2\pi)})$. If $x(n)$ is obtained by regular sampling of a

continuous-time signal $x_{ct}(t)$ such that $x(n) = x_{ct}(nT)$, where T is the sampling period, ω can be understood as the normalized frequency $\omega = 2\pi fT$.

The properties of the discrete-time Fourier transform are easily derived from those of the z-transform by choosing $z = e^{j\omega}$.

Shifting

$$x(n - n_0) \longleftrightarrow e^{-j\omega n_0} X(e^{j\omega}). \tag{4.27}$$

Convolution

$$x(n) * y(n) \longleftrightarrow X(e^{j\omega})\, Y(e^{j\omega}). \tag{4.28}$$

Multiplication in the Time Domain

$$x(n)y(n) \longleftrightarrow \frac{1}{2\pi} X(e^{j\omega}) * Y(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j(\omega - \nu)}) Y(e^{j\nu}) d\nu. \tag{4.29}$$

Reconstruction. If the sequence $x(n)$ is absolutely summable ($\boldsymbol{x} \in \ell_1(-\infty, \infty)$), it can be reconstructed from $X(e^{j\omega})$ via

$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega})\, e^{j\omega n}\, d\omega. \tag{4.30}$$

The expression (4.30) is nothing but the inverse z-transform, evaluated on the unit circle.

Parseval's Theorem. As in the case of continuous-time signals, the signal energy can be calculated in the time and frequency domains. If a signal $x(n)$ is absolutely and square summable ($\boldsymbol{x} \in \ell_1(-\infty, \infty) \cap \ell_2(-\infty, \infty)$), then

$$E_x = \sum_{n=-\infty}^{\infty} |x(n)|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |X(e^{j\omega})|^2\, d\omega. \tag{4.31}$$

Note that the expression (4.26) may be understood as a series expansion of the 2π-periodic spectrum $X(e^{j\omega})$, where the values $x(n)$ are the coefficients of the series expansion.

4.3 The Discrete Fourier Transform (DFT)

The transform pair of the *discrete Fourier transform* (DFT) is defined as

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{nk} \\ &\updownarrow \\ x(n) &= \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-nk}, \end{aligned} \tag{4.32}$$

where

$$W_N = e^{-j2\pi/N}. \tag{4.33}$$

Due to the periodicity of the basis functions, the DFT can be seen as the discrete-time Fourier transform of a periodic signal with period N.

With

$$\boldsymbol{x} = \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(N-1) \end{bmatrix} \tag{4.34}$$

and

$$\boldsymbol{W} = \left[W_N^{kn}\right] = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & W_N & \dots & W_N^{N-1} \\ \vdots & \vdots & & \vdots \\ 1 & W_N^{N-1} & \dots & W_N^{(N-1)(N-1)} \end{bmatrix} \tag{4.35}$$

the above relationships can also be expressed as

$$\boldsymbol{X} = \boldsymbol{W}\boldsymbol{x} \longleftrightarrow \boldsymbol{x} = \frac{1}{N}\boldsymbol{W}^H \boldsymbol{X}. \tag{4.36}$$

We see that $\boldsymbol{W}$ is orthogonal, but not orthonormal.

The DFT can be normalized as follows:

$$\boldsymbol{\alpha} = \boldsymbol{\Phi}^H \boldsymbol{x} \longleftrightarrow \boldsymbol{x} = \boldsymbol{\Phi}\boldsymbol{\alpha}, \tag{4.37}$$

where

$$\boldsymbol{\Phi} = \frac{1}{\sqrt{N}} \boldsymbol{W}^H. \tag{4.38}$$

The columns of $\boldsymbol{\Phi}$,

$$\boldsymbol{\varphi}_k = \frac{1}{\sqrt{N}} \left[1, W_N^{-k}, W_N^{-2k}, \dots, W_N^{-(N-1)k}\right]^T, \quad k = 0, 1, \dots, N-1$$

then form an orthonormal basis as usual.

We now briefly recall the most important properties of the DFT. For an elaborate discussion of applications of the DFT in digital signal processing the reader is referred to [113].

Shifting. A circular time shift by μ yields

$$\begin{aligned} x_\mu(n) &= x((n+\mu) \bmod N) \\ &\updownarrow \\ X_\mu(m) &= \sum_{n=0}^{N-1} x((n+\mu) \bmod N) W_N^{nm} \\ &= \sum_{i=0}^{N-1} x(i) W_N^{(i-\mu)m} \\ &= W_N^{-\mu m} X(m). \end{aligned} \tag{4.39}$$

Accordingly,

$$W_N^{kn}\, x(n) \;\longleftrightarrow\; \sum_{n=0}^{N-1} x(n) W_N^{n(m+k)} = X((m+k) \bmod N). \tag{4.40}$$

Multiplication and Circular Convolution. For the inverse discrete Fourier transform (IDFT) of

$$Y(m) = X_1(m)\, X_2(m)$$

we have

$$\begin{aligned} y(n) &= \frac{1}{N} \sum_{m=0}^{N-1} X_1(m)\, X_2(m)\, W_N^{-mn} \\ &= \frac{1}{N} \sum_{p=0}^{N-1} \sum_{q=0}^{N-1} x_1(p)\, x_2(q) \sum_{m=0}^{N-1} W_N^{-m(n-p-q)} \\ &= \sum_{p=0}^{N-1} x_1(p)\, x_2((n-p) \bmod N). \end{aligned} \tag{4.41}$$

That is, a multiplication in the frequency domain means a circular convolution

in the time domain. Accordingly,

$$x_1(n)\, x_2(n) \longleftrightarrow \sum_{n=0}^{N-1} X_1(n)\, X_2((m-n) \bmod N). \tag{4.42}$$

Complex Conjugation. Conjugation in the time or frequency domains yields

$$x^*(n) \longleftrightarrow X^*(N-n) \tag{4.43}$$

and

$$x^*(N-n) \longleftrightarrow X^*(n). \tag{4.44}$$

Relationship between the DFT and the KLT. The DFT is related to the KLT due to the fact that it diagonalizes any circulant matrix

$$\boldsymbol{H} = \begin{bmatrix} h_0 & h_{N-1} & \dots & h_1 \\ h_1 & h_0 & \dots & h_2 \\ \vdots & \vdots & & \vdots \\ h_{N-1} & h_{N-2} & \dots & h_0 \end{bmatrix}. \tag{4.45}$$

In order to show the diagonalization effect of the DFT, we consider a linear time-invariant system (FIR filter) with impulse response $h(n)$, $0 \le n \le N-1$ which is excited by the periodic signal $W_N^{-kn}/\sqrt{N}$. The output signal $y(n)$ is given by

$$y(n) = \frac{1}{\sqrt{N}} \underbrace{\left[\sum_{m=0}^{N-1} h(m) W_N^{mk}\right]}_{H(k)} W_N^{-nk} = \frac{1}{\sqrt{N}} H(k) W_N^{-nk}. \tag{4.46}$$

Using the property $W_N^{n+N} = W_N^n$, we get

$$\begin{aligned} \begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-1) \end{bmatrix} &= \frac{H(k)}{\sqrt{N}} \begin{bmatrix} 1 \\ W_N^{-k} \\ \vdots \\ W_N^{-k(N-1)} \end{bmatrix} \\ &= \frac{1}{\sqrt{N}} \begin{bmatrix} h_0 & h_{N-1} & \dots & h_1 \\ h_1 & h_0 & \dots & h_2 \\ \vdots & \vdots & & \vdots \\ h_{N-1} & h_{N-2} & \dots & h_0 \end{bmatrix} \begin{bmatrix} 1 \\ W_N^{-k} \\ \vdots \\ W_N^{-k(N-1)} \end{bmatrix} \end{aligned} \tag{4.47}$$

Comparing (4.47) with (4.45) and (4.37) yields the relationship

$$H(k)\boldsymbol{\varphi}_k = \boldsymbol{H}\boldsymbol{\varphi}_k, \quad k=0,1,\ldots,N-1. \tag{4.48}$$

Thus, the eigenvalues $\lambda_k = H(k)$ of $\boldsymbol{H}$ are derived from the DFT of the first column of $\boldsymbol{H}$. The vectors $\boldsymbol{\varphi}_k$, $k=0,1,\ldots,N-1$ are the eigenvectors of $\boldsymbol{H}$. We have

$$\boldsymbol{\Phi}^H \boldsymbol{H} \boldsymbol{\Phi} = \operatorname{diag}\{H(0), H(1), \ldots, H(N-1)\}. \tag{4.49}$$

4.4 The Fast Fourier Transform

For a complex-valued input signal $x(n)$ of length N the implementation of the DFT matrix $\boldsymbol{W}$ requires N^2 complex multiplications. The idea behind the fast Fourier transform (FFT) is to factorize $\boldsymbol{W}$ into a product of sparse matrices that altogether require a lower implementation cost than the direct DFT. Thus, the FFT is a fast implementation of the DFT rather than a different transform with different properties.

Several concepts for the factorization of $\boldsymbol{W}$ have been proposed in the literature. We will mainly focus on the case where the DFT length is a power of two. In particular, we will discuss the radix-2 FFTs, the radix-4 FFT, and the split-radix FFT. Section 4.4.5 gives a brief overview of FFT algorithms for cases where N is not a power of two.

We will only discuss the forward DFT. For the inverse DFT a similar algorithm can be found.

4.4.1 Radix-2 Decimation-in-Time FFT

Let us consider an N-point DFT where N is a power of two, i.e. $N = 2^K$ for some $K \in \mathbb{N}$. The first step towards a fast implementation is to decompose the time signal $x(n)$ into its even and odd numbered components

$$\left.\begin{aligned} u(n) &= x(2n) \\ v(n) &= x(2n+1) \end{aligned}\right\} \quad n = 0,1,\ldots,\frac{N}{2}-1. \tag{4.50}$$

The DFT can be written as

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{nk} \\ &= \sum_{n=0}^{\frac{N}{2}-1} u(n) W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} v(n) W_N^{(2n+1)k} \\ &= \sum_{n=0}^{\frac{N}{2}-1} u(n) W_{N/2}^{nk} \quad + \quad W_N^k \sum_{n=0}^{\frac{N}{2}-1} v(n) W_{N/2}^{nk}, \qquad k = 0, 1, \ldots, N-1. \end{aligned} \tag{4.51}$$

In the last step the properties $W_N^{2nk} = W_{N/2}^{nk}$ and $W_N^{(2n+1)k} = W_N^k W_{N/2}^{nk}$ were used. The next step is to write (4.51) for $k = 0, 1, \ldots, \frac{N}{2} - 1$ as

$$X(k) = U(k) + W_N^k V(k), \quad k = 0, 1, \ldots, \frac{N}{2} - 1 \tag{4.52}$$

with

$$\begin{aligned} U(k) &= \sum_{n=0}^{\frac{N}{2}-1} u(n) W_{N/2}^{nk}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1 \\ V(k) &= \sum_{n=0}^{\frac{N}{2}-1} v(n) W_{N/2}^{nk}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1. \end{aligned} \tag{4.53}$$

Due to the periodicity of the DFT the values $X(k)$ for $k = \frac{N}{2}, \ldots, N-1$ are given by

$$X(k) = U(k - \frac{N}{2}) + W_N^k V(k - \frac{N}{2}), \quad k = \frac{N}{2}, \ldots, N-1. \tag{4.54}$$

Thus, we have decomposed an N-point DFT into two $\frac{N}{2}$-point DFTs and some extra operations for combining the two DFT outputs.

Figure 4.1 illustrates the implementation of an N-point DFT via two $\frac{N}{2}$-point DFTs. It is easily verified that the decomposition of the DFT results in a reduction of the number of multiplications: the two DFTs require $2(N/2)^2$ multiplications, and the prefactors W_N^k require another N multiplications. Thus, the overall complexity is $\frac{N^2}{2} + N$, instead of N^2 for the direct DFT. The prefactors W_N^k, which are used for the combination of the two DFTs, are called *twiddle factors*.

Since N is considered to be a power of two, the same decomposition principle can be used for the smaller DFTs and the complexity can be further

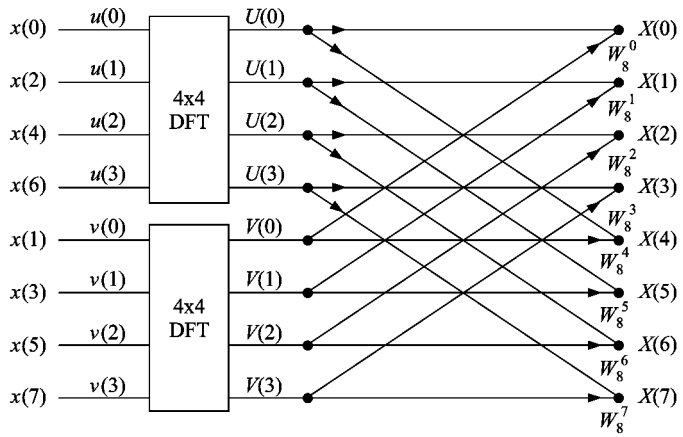


Figure 4.1. Realization of an 8-point DFT via two 4-point DFTs.

reduced. To be explicit, we decompose the sequences $u(n)$ and $v(n)$ into their even and odd numbered parts:

$$\begin{array}{rcccl} a(n) & = & u(2n) & = & x(4n) \\ b(n) & = & u(2n+1) & = & x(4n+2) \\ c(n) & = & v(2n) & = & x(4n+1) \\ d(n) & = & v(2n+1) & = & x(4n+3). \end{array} \tag{4.55}$$

Observing that $W_{N/2}^{k} = W_N^{2k}$ we get for the $\frac{N}{2}$-point DFTs $U(k)$ and $V(k)$

$$U(k) = \begin{cases} A(k) + W_N^{2k}\, B(k), & k = 0, 1, \ldots, \dfrac{N}{4} - 1 \\ A(k - \frac{N}{4}) + W_N^{2k}\, B(k - \frac{N}{4}), & k = \dfrac{N}{4}, \ldots, \dfrac{N}{2} - 1 \end{cases} \tag{4.56}$$

and

$$V(k) = \begin{cases} C(k) + W_N^{2k}\, D(k), & k = 0, 1, \ldots, \dfrac{N}{4} - 1 \\ C(k - \dfrac{N}{4}) + W_N^{2k}\, D(k - \dfrac{N}{4}), & k = \dfrac{N}{4}, \ldots, \dfrac{N}{2} - 1. \end{cases} \tag{4.57}$$

The decomposition procedure can be continued until two-point DFTs are reached.

It turns out that all stages of the FFT are composed of so-called *butterfly graphs* as shown in Figure 4.2. The two structures in Figure 4.2 are equivalent,

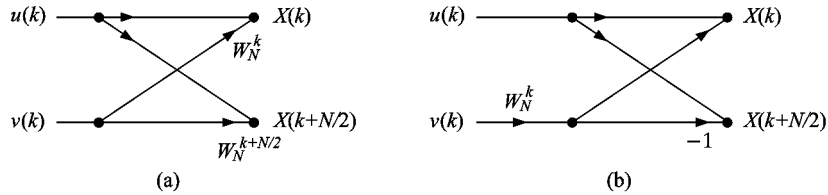


Figure 4.2. Equivalent butterfly graphs.

but the one in Figure 4.2(b) saves us one complex multiplication. The complete flow graph for an 8-point FFT based on the butterfly in Figure 4.2(b) is depicted in Figure 4.3. As we see, the output values appear in their natural order, but the input values appear in permuted order. This is the case for all N. The order of the input values is known as the *bit reversed order*. This order can be derived from the natural one as follows. First, one represents the numbers $0, 1, \ldots, N-1$ in binary form. Then the order of bits is reversed and the decimal equivalent is taken. For example, $n = 3$ is represented by [011] when an 8-point FFT is considered. This yields [110] in reversed order, and the decimal equivalent is 6. Thus, $x(6)$ has to be connected to input node 3.

Since the butterfly operations within each stage of the FFT are independent of one another, the computation of the FFT can be carried out *in place*. This means that the pair of output values of a butterfly is written over the input. After this has been done for all butterflies of a given processing stage, one can proceed to the next stage. Thus, only a memory of size $N + 1$ is required for computing an N-point FFT.

The computational complexity of the FFT is as follows. Each stage of the FFT requires $N/2$ complex multiplications and N additions. The number of stages is $\log_2 N$. This yields a total number of $\frac{1}{2} N \log_2 N$ complex multiplications and $N \log_2 N$ additions. However, since the 2-point DFTs do not require multiplications, and since the 4-point DFTs involve multiplications with $1, -1, j$, and $-j$ only, the actual number of full complex multiplications is even lower than $\frac{1}{2} N \log_2 N$.

4.4.2 Radix-2 Decimation-in-Frequency FFT

A second variant of the radix-2 FFT is the *decimation-in-frequency* algorithm. In order to derive this algorithm, we split the input sequence into the first

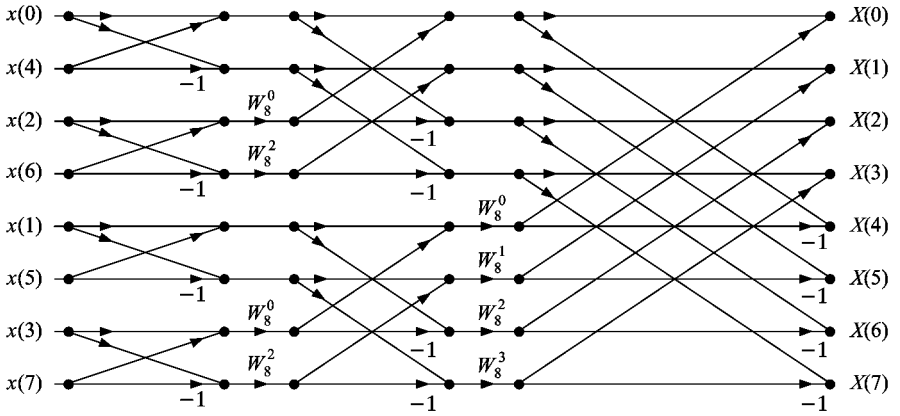


Figure 4.3. Flow graph for an 8-point decimation-in-time FFT.

and second halves and write the DFT as

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{nk} \\ &= \sum_{n=0}^{N/2-1} u(n) W_N^{nk} + v(n) W_N^{(n+N/2)k} \\ &= \sum_{n=0}^{N/2-1} [u(n) + (-1)^k v(n)]\, W_N^{nk}, \end{aligned} \tag{4.58}$$

where

$$\left.\begin{aligned} u(n) &= x(n) \\ v(n) &= x(n+N/2) \end{aligned}\right\}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.59}$$

In (4.58) we have used the fact that $W_N^{N/2} = -1$. For the even and odd numbered DFT points we get

$$X(2k) = \sum_{n=0}^{N-1} [u(n) + v(n)]\, W_N^{2nk} \tag{4.60}$$

and

$$X(2k+1) = \sum_{n=0}^{N-1} [u(n) - v(n)]\, W_N^n\, W_N^{2nk}. \tag{4.61}$$

Because of $W_N^{2nk} = W_{N/2}^{nk}$, the even numbered DFT points $X(2k)$ turn out to be the DFT of the half-length sequence $u(n) + v(n)$. The odd numbered

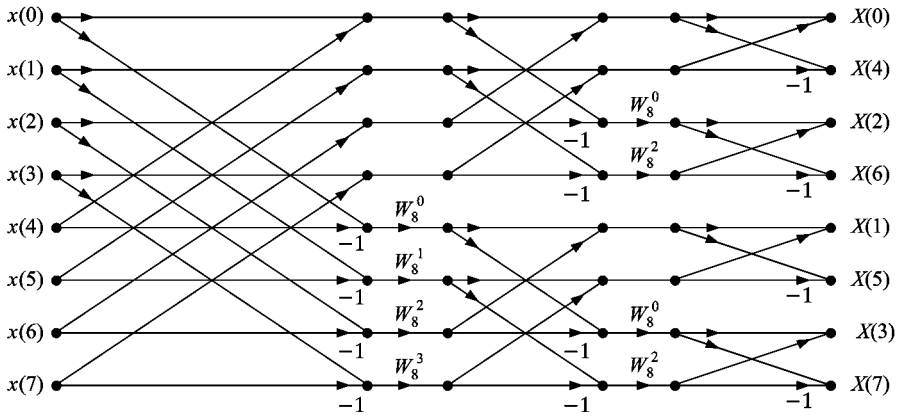


Figure 4.4. Flow graph for an 8-point decimation-in-frequency FFT.

DFT points $X(2k+1)$ are the DFT of $[u(n) - v(n)]\, W_N^n$. Thus, as with the decimation-in-time algorithm, the N-point DFT is decomposed into two $N/2$-point DFTs. Using the principle repeatedly results in an FFT algorithm where the input values appear in their natural order, but where the output values appear in bit reversed order. The complexity is the same as for the decimation-in-time FFT. Figure 4.4 shows the complete flow graph of the decimation-in-frequency FFT for the case $N = 8$. The comparison of Figures 4.3 and 4.4 shows that the two graphs can be viewed as transposed versions of one another.

4.4.3 Radix-4 FFT

The radix-4 decimation-in-frequency FFT is derived by writing the DFT as

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{nk} \\ &= \sum_{n=0}^{N/4-1} \left[\sum_{\ell=0}^{3} x(n+\ell\frac{N}{4})\, W_N^{(N/4)\ell k} \right] W_N^{nk} \\ &= \sum_{n=0}^{N/4-1} \left[\sum_{\ell=0}^{3} x(n+\ell\frac{N}{4})\, (-j)^{\ell k} \right] W_N^{nk}. \end{aligned} \tag{4.62}$$

Splitting $X(k)$ into four subsequences $X(4k+m)$ yields

$$X(4k+m) = \sum_{n=0}^{N/4-1} \left[\sum_{\ell=0}^{3} (-j)^{\ell m} x(n+\ell\frac{N}{4}) W_N^{nm} \right] W_{N/4}^{nk}. \tag{4.63}$$

Thus, we have replaced the computation of an N-point DFT by four $N/4$-point DFTs. One of these four DFTs requires no multiplication at all, and the others require one complex multiplication per point. Compared to the radix-2 FFT this means $3 \times (N/4)$ instead of $N/2$ multiplications for the twiddle factors. However, the radix-4 algorithm requires only $N/4$-point DFTs, and it requires only half as many stages as a radix-2 one. Therefore, the overall number of multiplications is lower for the radix-4 case.

4.4.4 Split-Radix FFT

The split-radix FFT [46], which is a mixture of the radix-2 and radix-4 algorithm, requires the lowest number of operations of all currently known FFT algorithms. It is also easily programmed on a computer. The radix-2 approach is used to compute the even numbered frequencies, and the radix-4 approach is used to compute two length-$(N/4)$ subsequences of the odd numbered frequencies. For this, $X(k)$ is split into the following three subsets:

$$X(2k) = \sum_{n=0}^{N/2-1} [x(n) + x(n+\frac{N}{2})]\, W_{N/2}^{nk}, \tag{4.64}$$

$$X(4k+1) = \sum_{n=0}^{N/4-1} \Big[[x(n) - x(n+\frac{N}{2})] - j[x(n+\frac{N}{4}) - x(n+\frac{3N}{4})] \Big] W_N^n W_{N/4}^{nk}, \tag{4.65}$$

$$X(4k+3) = \sum_{n=0}^{N/4-1} \Big[[x(n) - x(n+\frac{N}{2})] + j[x(n+\frac{N}{4}) - x(n+3\frac{N}{4})] \Big] W_N^{3n} W_{N/4}^{nk}. \tag{4.66}$$

The terms $[x(n) - x(n+\frac{N}{2})]$ and $[x(n+\frac{N}{4}) - x(n+\frac{3N}{4})]$ in (4.65) and (4.66) are the natural pairs to the terms in (4.64). Thus, a split-radix butterfly can be drawn as shown in Figure 4.5. As with the previous approaches, the decomposition principle can be used repeatedly. It turns out that the split-radix approach requires less multiplications than a pure radix-2 or radix-4 FFT, because fewer full complex multiplications occur. The split-radix

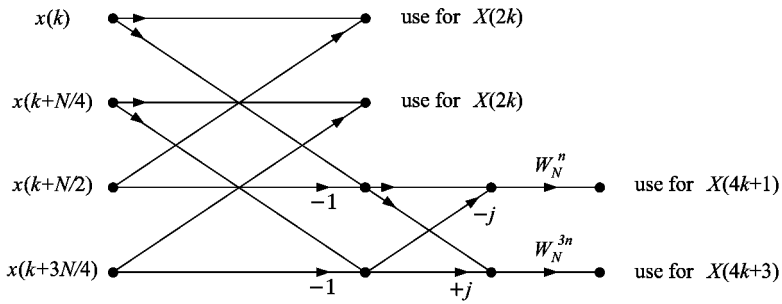


Figure 4.5. Butterfly for a split-radix FFT.

concept can be generalized to other radices [152], and special forms are available for real and real-symmetric data [45, 138].

4.4.5 Further FFT Algorithms

There are a number of algorithms available for the case where the DFT length is not necessarily a power of two. The best known one is the *Cooley–Tukey FFT* [31], which requires that the DFT-length is a composite number $N = PQ$, where P and Q are integers. The DFT can then be written as

$$\begin{aligned} X(kP+m) &= \sum_{i=0}^{P-1} \sum_{j=0}^{Q-1} x(iQ+j)\, W_N^{(iQ+j)(kP+m)} \\ &= \sum_{j=0}^{Q-1} W_Q^{jk}\, W_N^{jm} \sum_{i=0}^{P-1} x(iQ+j)\, W_P^{im} \end{aligned} \tag{4.67}$$

for $k = 0, 1, \ldots, P-1$ and $m = 0, 1, \ldots, Q-1$. The inner sum in the second line of (4.67) turns out to be a P-point DFT, and the outer sum is a Q-point DFT. Thus, the N-point DFT is decomposed into P Q-point and Q P-point DFTs, plus the twiddle factors in the middle of the second line in (4.67). As can easily be verified, the complexity is lower than for the direct N-point DFT. If P and/or Q are composite themselves, the approach can be iterated, and the complexity can be further reduced. Note that the radix-2 approach occurs as a special case where $P = 2$ and $Q = N/2$.

If the DFT-length can be factored into $N = PQ$ where P and Q are relatively prime (have no common divisor other than 1) a powerful algorithm known as the *Good–Thomas FFT* can be used. The basic idea dates back to papers by Good [64] and Thomas [143]. The algorithm has been further developed in [88, 164, 20, 142]. The efficiency of the Good–Thomas FFT

results from the fact that for relatively prime P and Q the twiddle factors (they are always present in the Cooley–Tukey FFT) can be avoided. The input data can be arranged in a two-dimensional array, and the transform can be implemented as a true two-dimensional transform. The mapping is based on the Chinese remainder theorem [9].

FFTs where N is a prime number can be realized via circular convolution [120, 10]. In order to give an idea of how this is done, we follow the approach in [10] and write the DFT as

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} = W_{2N}^{k^2} \sum_{n=0}^{N-1} [x(n) W_{2N}^{n^2}] \, W_{2N}^{-(k-n)^2}. \tag{4.68}$$

The sum on the right side can be identified as a circular convolution of the sequences $x(n)W_{2N}^{n^2}$ and $W_{2N}^{-n^2}$, that is

$$X(n) = W_{2N}^{k^2} \, [x(n) W_{2N}^{n^2} * W_{2N}^{-n^2}]. \tag{4.69}$$

Efficiency is achieved by implementing the circular convolution via fast convolution based on the FFT, see e.g. [117].

Powerful FFT algorithms are most often associated with signal lengths that are powers of two. However, prime factor algorithms such as the Winograd FFT [164] are often competitive, if not superior, to the power-of-two approaches. Thus, when designing an algorithm where the DFT is involved, one should not be bound to certain block lengths, because for almost any length an appropriate FFT algorithm can be found.

4.5 Discrete Cosine Transforms

We distinguish the following four types of *discrete cosine transforms* (DCTs) [122]:

DCT-I:

$$c_k^I(n) = \sqrt{\frac{2}{N}} \, \gamma_k \gamma_n \, \cos\left(\frac{kn\pi}{N}\right), \qquad k, n = 0, 1, \ldots, N. \tag{4.70}$$

DCT-II:

$$c_k^{II}(n) = \sqrt{\frac{2}{N}} \, \gamma_k \, \cos\left(\frac{k(n+\frac{1}{2})\pi}{N}\right), \qquad k, n = 0, 1, \ldots, N-1. \tag{4.71}$$

DCT-III:

$$c_k^{III}(n) = \sqrt{\frac{2}{N}}\ \gamma_n\ \cos\left(\frac{(k+\frac{1}{2})n\pi}{N}\right), \qquad k,n = 0,1,\ldots,N-1. \tag{4.72}$$

DCT-IV:

$$c_k^{IV}(n) = \sqrt{\frac{2}{N}}\ \cos\left(\frac{(k+\frac{1}{2})(n+\frac{1}{2})\pi}{N}\right), \qquad k,n = 0,1,\ldots,N-1. \tag{4.73}$$

The constants γ_j in (4.70) – (4.72) are given by

$$\gamma_j = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } j = 0 \text{ or } j = N, \\ 1 & \text{otherwise.} \end{cases} \tag{4.74}$$

The coefficients $c_k(n)$ are the elements of the orthonormal basis vectors

$$\boldsymbol{c}_k(n) = [c_k(0), c_k(1), \ldots]^T.$$

In order to point out clearly how (4.70) – (4.73) are to be understood, let us consider the forward and inverse DCT-II:

$$\begin{aligned} X_C^{II}(k) &= \sum_{n=0}^{N-1} x(n)\, c_k^{II}(n) \\ &= \gamma_k \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x(n)\ \cos\left(\frac{k(n+\frac{1}{2})\pi}{N}\right), \end{aligned} \tag{4.75}$$

and

$$\begin{aligned} x(n) &= \sum_{k=0}^{N-1} X_C^{II}(k)\, c_k^{II}(n) \\ &= \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} X_C^{II}(k)\ \gamma_k\ \cos\left(\frac{k(n+\frac{1}{2})\pi}{N}\right). \end{aligned} \tag{4.76}$$

Especially the DCT-II is of major importance in signal coding because it is close to the KLT for first-order autoregressive processes with a correlation coefficient that is close to one.[1] To illustrate this, we consider the inverse of the correlation matrix of an AR(1) process, which is given by

[1] See Section 5.3 for the definition of an AR process.

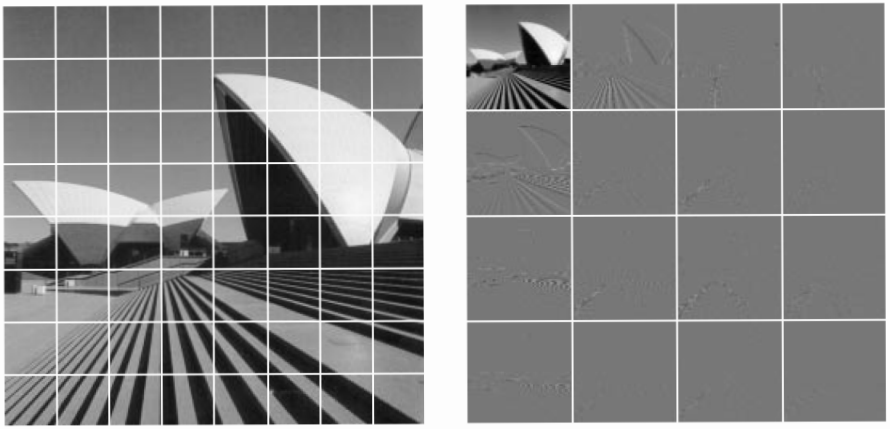
$$\boldsymbol{R}_{xx}^{-1} = \frac{1+\rho^2}{\sigma^2(1-\rho^2)} \begin{bmatrix} (1-\rho\beta) & -\beta & & & & \\ -\beta & 1 & -\beta & & & \\ & -\beta & \ddots & \ddots & & \\ & & & 1 & -\beta & \\ & & & -\beta & (1-\rho\beta) \end{bmatrix} \tag{4.77}$$

with $\beta = \rho/(1+\rho^2)$. The basis vectors of the DCT-II are the eigenvectors of tridiagonal symmetric matrices of the form

$$\boldsymbol{Q} = \begin{bmatrix} (1-\alpha) & -\alpha & & & & \\ -\alpha & 1 & -\alpha & & & \\ & -\alpha & \ddots & \ddots & & \\ & & & 1 & -\alpha & \\ & & & -\alpha & (1-\alpha) \end{bmatrix}. \tag{4.78}$$

We see that $\boldsymbol{Q}$ approaches $\boldsymbol{R}_{xx}^{-1}$ for $\rho \to 1$. Since the eigenvectors of $\boldsymbol{R}_{xx}$ are equal to those of $\boldsymbol{R}_{xx}^{-1}$ the DCT-II approaches the KLT for $\rho \to 1$. This means that the DCT-II has good decorrelation properties when the process which is to be transformed has high correlation ($\rho \to 1$). This is the case for most images, which explains why most image coding standards (e.g. JPEG, MPEG [79, 157, 108, 56]) are based on the DCT-II. Compared to the KLT, the DCT-II has the advantage that fast implementations based on the FFT algorithm exist [122].

Application in Image Coding. In most standards for transform coding of images, the two-dimensional cosine transform is used [79, 157, 108, 56]. Figure 4.6 gives an example. First, the two-dimensional signal is decomposed into non-overlapping blocks. Each of these blocks is then transformed separately. This operation can be written as $\boldsymbol{Y}_{N\times N} = \boldsymbol{U}^T\, \boldsymbol{X}_{N\times N}\, \boldsymbol{U}$, where $\boldsymbol{X}_{N\times N}$ is such a signal block and $\boldsymbol{U}$ is the DCT-II transform matrix whose columns contain the basis vectors of the DCT-II. Instead of the original $\boldsymbol{X}$, the representation $\boldsymbol{Y}$ is quantized and coded. From the quantized representation $\boldsymbol{Y}' = Q(\boldsymbol{Y})$ an approximation of the original is finally reconstructed. In Figure 4.6 we see that most of the energy of the transformed signal is concentrated in the top left sub-image. Such a concentration of signal energy in a few coefficients is the key to efficient compression. If we were to simply transmit the top left sub-image and neglect the others, we already could achieve drastic compression, while the reconstructed signal would still be relatively close to the original.



(a) (b)

Figure 4.6. Transform coding of images; (a) original, divided into $N \times N$ blocks; (b) transformed image after rearranging the pixels.

4.6 Discrete Sine Transforms

The *discrete sine transforms (DSTs)* are classified as follows [122]:

DST-I:

$$s_k^I(n) = \sqrt{\frac{2}{N}} \sin\left(\frac{kn\pi}{N}\right), \qquad k,n = 1,2,\ldots,N-1. \tag{4.79}$$

DST-II:

$$s_k^{II}(n) = \sqrt{\frac{2}{N}}\, \gamma_{k+1} \sin\left(\frac{(k+1)\,(n+\frac{1}{2})\pi}{N}\right), \qquad k,n = 0,1,\ldots,N-1. \tag{4.80}$$

DST-III:

$$s_k^{III}(n) = \sqrt{\frac{2}{N}}\, \gamma_{n+1} \sin\left(\frac{(k+\frac{1}{2})\,(n+1)\pi}{N}\right), \qquad k,n = 0,1,\ldots,N-1. \tag{4.81}$$

DST-IV:

$$s_k^{IV}(n) = \sqrt{\frac{2}{N}} \sin\left(\frac{(k+\frac{1}{2})(n+\frac{1}{2})\pi}{N}\right), \qquad k,n = 0,1,\ldots,N-1. \tag{4.82}$$

The constants γ_j in (4.79) – (4.81) are

$$\gamma_j = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } j = 0 \text{ or } j = N, \\ 1 & \text{otherwise.} \end{cases} \tag{4.83}$$

To be explicit, the forward and the inverse DST-II are given by

$$\begin{aligned} X_S^{II}(k) &= \sum_{n=0}^{N-1} x(n)\, s_k^{II}(n) \\ &= \gamma_{k+1} \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x(n) \ \sin\left(\frac{(k+1)\,(n+\frac{1}{2})\pi}{N}\right), \end{aligned} \tag{4.84}$$

and

$$\begin{aligned} x(n) &= \sum_{k=0}^{N-1} X_S^{II}(k)\, s_k^{II}(n) \\ &= \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} X_S^{II}(k)\ \gamma_{k+1}\ \sin\left(\frac{(k+1)\,(n+\frac{1}{2})\pi}{N}\right). \end{aligned} \tag{4.85}$$

The DST-II is related to the KLT by the fact that the KLT for an AR(1) process with correlation coefficient $\gamma \to -1$ approaches the DST-II. Thus, the DST-II has good compaction properties for processes with negative correlation of adjacent samples.

4.7 The Discrete Hartley Transform

The Hartley transform as discussed in Section 2.3 received little attention until its discrete version, the *discrete Hartley transform (DHT)*, was introduced in the early 1980s by Wang [158, 159, 160] and Bracewell [13, 14, 15]. Like other discrete transforms such as the DFT or the DCT, the DHT can be implemented efficiently through a factorization of the transform matrix. This results in fast algorithms that are closely related to the FFT, and in fact, the *fast Hartley transform (FHT)* can be computed via the FFT, and vice versa, the FFT can be implemented via the FHT [161, 14, 139]. For example, in [139] a split-radix approach for the FHT has been proposed.

The forward and inverse discrete Hartley transform pair is given by

$$\begin{aligned} X_H(k) &= \sum_{n=0}^{N-1} x(n)\ \mathrm{cas}\,\frac{2\pi nk}{N} \\ &\updownarrow \\ x(n) &= \frac{1}{N} \sum_{k=0}^{N-1} X_H(k)\ \mathrm{cas}\,\frac{2\pi nk}{N}, \end{aligned} \tag{4.86}$$

where $\operatorname{cas}\phi = \cos\phi + \sin\phi$. The signal $x(t)$ is considered to be real-valued, so that also the transform is real-valued. As with the DFT, the sequence $X_H(k)$ is periodic with period N.

Note that apart from the prefactor $1/N$ the DHT is self-inverse, which means that the same computer program or hardware can be used for the forward and inverse transform. This is not the case for the DFT, where a real-valued signal is transformed into a complex spectrum.

We may interpret the basis sequences $\operatorname{cas}(2\pi nk/N)$ as sample values of the basis functions $\operatorname{cas}\omega_k t$ with $\omega_k = 2\pi k/N$. The basis function with the highest frequency then occurs for $k = N/2$. The kth and the $(N-k)$th frequency are the same.

The relationships between the DHT and the DFT are easily derived. Using the fact that

$$e^{j\phi} = \frac{1+j}{2}\operatorname{cas}\phi + \frac{1-j}{2}\operatorname{cas}(-\phi) \tag{4.87}$$

and the periodicity in N, we get

$$\begin{aligned} \Re\{X(k)\} &= \frac{X_H(k) + X_H(N-k)}{2}, \\ \Im\{X(k)\} &= -\frac{X_H(k) - X_H(N-k)}{2}, \end{aligned} \tag{4.88}$$

where $X(k)$ denotes the DFT. The DHT can be expressed in terms of the DFT as

$$X_H(k) = \Re\{X(k)\} - \Im\{X(k)\}. \tag{4.89}$$

The properties of the DHT can easily be derived from the definition (4.86). Like in the continuous-time case, most of them are very similar to the properties of the Fourier transform. We will briefly discuss the most important ones. The proofs are essentially the same as for the continuous-time case and are omitted here.

Time Inversion. From (4.86) we see that

$$x(N-n) \longleftrightarrow X_H(N-n). \tag{4.90}$$

Shifting. A circular time shift by μ yields

$$\begin{gathered} x((n+\mu) \bmod N) \\ \updownarrow \\ \cos\left(\frac{2\pi\mu k}{N}\right) X_H(k) + \sin\left(\frac{2\pi\mu k}{N}\right) X_H(N-k). \end{gathered} \tag{4.91}$$

Circular Convolution. The correspondence for a circular convolution of two time sequences $x(n)$ and $y(n)$ is

$$\begin{aligned} &\sum_{p=0}^{N-1} x(p)\, y((n-p) \bmod N) \\ &\qquad\qquad \updownarrow \\ &\frac{N}{2}\,[\, X_H(k)\, Y_H(k) - X_H(N-k)\, Y_H(N-k) \\ &\qquad + X_H(k)\, Y_H(N-k) \;+ X_H(N-k)\, Y_H(k)\,]. \end{aligned} \tag{4.92}$$

Multiplication. The correspondence for products $x(n)y(n)$ is

$$\begin{aligned} x(n)\, y(n) \longleftrightarrow \frac{N}{2}[\, &X_H(k) * Y_H(k) + X_H(-k) * Y_H(k) \\ &+ X_H(k) * Y_H(-k) - X_H(-k) * Y_H(-k)\,], \end{aligned} \tag{4.93}$$

where the convolutions have to be carried out in a circular manner. For example an expression $Z_H(k) = X_H(k) * Y_H(-k)$ means

$$Z_H(k) = \sum_{p=0}^{N-1} X_H(p)\, Y_H((k-p) \bmod N). \tag{4.94}$$

Remarks. The question of whether the DFT or the DHT should be used in an application very much depends on the application itself. As mentioned earlier, fast algorithms exist for both transforms, and one fast transform can be used in order to implement the other transform in an efficient way. For both the FFT and the FHT the complexity is $N \log_2 N$. An advantage of the DHT is the fact that the DHT is self-inverse, so that only one software routine or hardware device is needed for the forward and inverse FHT. For the forward and inverse FFT of a real signal, two different routines or devices are required. The DHT is somehow conceptually simpler than the DFT if the input signal is real, but all operations can be carried out with the FFT and the FHT with the same complexity.

4.8 The Hadamard and Walsh–Hadamard Transforms

The basis vectors of the *discrete Hadamard* and the *discrete Walsh–Hadamard transforms* consist of the values $\pm\alpha$; just like the Walsh functions discussed in Section 3.2.6. Both transforms are unitary. Basically they differ only in the order of the basis vectors.

We have

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{H}\,\boldsymbol{x}, \\ \boldsymbol{x} &= \boldsymbol{H}\,\boldsymbol{y}, \end{aligned} \tag{4.95}$$

where $\boldsymbol{x}$ denotes the signal, $\boldsymbol{y}$ the representation, and $\boldsymbol{H}$ the transform matrix of the Hadamard transform. $\boldsymbol{H}$ is symmetric and self-inverse:

$$\boldsymbol{H}^T = \boldsymbol{H} = \boldsymbol{H}^{-1}. \tag{4.96}$$

The transform matrix of the 2×2-Hadamard transform is given by

$$H^{(2)} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{4.97}$$

From this, all transform matrices $\boldsymbol{H}^{(n)}$ of size[2] $n = 2^k$, $k \in \mathbb{N}$ can be calculated recursively [133]:

$$\boldsymbol{H}^{(2n)} = \frac{1}{\sqrt{2}} \begin{bmatrix} \boldsymbol{H}^{(n)} & \boldsymbol{H}^{(n)} \\ \boldsymbol{H}^{(n)} & -\boldsymbol{H}^{(n)} \end{bmatrix}. \tag{4.98}$$

The *Walsh–Hadamard transform* is obtained by taking the Hadamard transform and rearranging the basis vectors according to the number of zero crossings [66]. Somehow, this yields an order of the basis vectors with respect to their spectral properties.

[2] There exist Hadamard matrices whose dimension is not a power of two.